Projekt 2 - Práce s datovými strukturami

Motivace projektu

V prvním projektu jsme si vyzkoušeli práci s textem a omezeným tabulkovým procesorem. Hlavní omezení spočívalo v sekvenční úpravě vstupních dat. V tomto projektu budeme chtít vytvořit tabulkový procesor, který dokáže pracovat s daty v celé tabulce. Navíc tabulkový procesor nebude mít omezení obsahu buněk nebo počtu buněk v tabulce.

Popis projektu

Cílem projektu je vytvořit program, který bude implementovat základní operace tabulkových procesorů. Vstupem programu budou textová data tabulky, zadání operací bude prostřednictvím argumentů příkazové řádky a výsledek zpracování tabulky bude program ukládat do zadaného souboru.

Detailní specifikace

Program implementujte ve zdrojovém souboru *sps.c* (SPreadSheet). Vstupní data budou čtena ze souboru, jehož jméno bude zadáno na příkazové řádce. Program provádí operace zadané "příkazy tabulkového procesoru", které provádí v sekvenci zadané na příkazové řádce.

Překlad a odevzdání zdrojového souboru

Odevzdání: Odevzdejte zdrojový soubor *sps.c* prostřednictvím informačního systému.

Překlad: Program překládejte s následujícími argumenty

```
$ gcc -std=c99 -Wall -Wextra -Werror sps.c -o sps
```

Syntax spuštění

Program se spouští v následující podobě: (./sps značí umístění a název programu):

```
./sps [-d DELIM] CMD_SEQUENCE FILE
```

- Argument `-d` specifikuje, jaké znaky lze interpretovat jako oddělovače jednotlivých buněk. Každý znak ve vstupním řádku, který je obsažen v řetězci DELIM, se chová jako oddělovač dvou sousedících buněk. Ve výchozím nastavení je DELIM řetězec obsahující mezeru. Vícenásobný výskyt stejných znaků v řetězci DELIM je ignorován. První znak z řetězce DELIM bude také použit jako oddělovač výstupních hodnot. DELIM nesmí obsahovat uvozovky ani zpětné lomítko. Argument -d DELIM je volitelný.
- Argument `CMD_SEQUENCE` je jeden argument obsahující sekvenci příkazů. Více příkazů tabulkového procesoru je odděleno středníkem. Příkaz nesmí být prázdný.
- Argument `FILE` specifikuje název souboru s tabulkou.

- Volitelně (usnadní pokročilý vývoj projektu, není třeba pro získání bodového hodnocení): Příkazy mohou být uloženy v textovém souboru. Každý příkaz bude oddělen znakem konce řádku. Soubor s příkazy bude specifikován namísto sekvence příkazů jako argument: `-cCOMMAND_FILE`.

Formát tabulky

Tabulka je textový soubor obsahující řádky tabulky (každý řádek je ukončen znakem konce řádku). Každý řádek obsahuje jednotlivé buňky, které jsou oddělené právě jedním znakem ze sady oddělovačů. Data v buňce jsou textová (celá čísla a čísla s plovoucí řádovou čárkou jsou v desítkové soustavě a jsou kompatibilní se zápisem v jazyku C). Textové řetězce mohou být ohraničené uvozovkami (na začátku a na konci buňky), mohou také obsahovat uvozené znaky (tzv. Escape sekvence) pomocí zpětného lomítka: `\X` reprezentuje znak X, kde X může být libovolný znak (kromě znaku konce řádku).

Příkazy tabulkového procesoru

Příkaz tabulkového procesoru se vždy uplatňuje pro vybrané buňky. Vybrané buňky mohou být buď samostatné buňky nebo celý řádek, celý sloupec nebo tzv. okno buněk. Každá buňka je identifikována indexem řádku a sloupce, počítáno od 1. Počáteční výběr je první buňka v tabulce (řádek 1, sloupec 1). Nový příkaz výběru buněk vždy přenastaví předchozí výběr. Pro daný výběr mohou být prováděné různé příkazy. Příkazy se liší podle toho, zda pracují nad jednotlivými buňkami (v takovém případě se příkaz aplikuje postupně po řádcích a v každém řádku po sloupcích - tj. stejně, jak jsme zvyklí číst text), zda pracují nad celým sloupcem (pak se aplikují pouze jednou pro každý sloupec výběru, s rostoucím indexem), nebo zda pracují nad celým řádkem (potom se aplikují pouze jednou pro každý řádek výběru, s rostoucím indexem).

Vybrané příkazy mohou akceptovat parametr jako řetězec označovaný STR. Na takový řetězec se vztahují stejná pravidla jako na textový obsah buňky specifikovaný v odstavci popisující formát tabulky (tj. může být v uvozovkách, může obsahovat speciální znaky uvozené zpětným lomítkem).

Příkazy pro změnu výběru

Výběr buněk může přesahovat hranice vstupní tabulky. V takovém případě bude tabulka rozšířena o požadovaný počet řádků a sloupců obsahující prázdné buňky až do aktuálního výběru.

- `[R,C]` - výběr buňky na řádku R a sloupci C.
- `[R,_]` - výběr celého řádku R.
- `[_,C]` - výběr celého sloupce C.
- `[R1,C1,R2,C2]` - výběr okna, tj. všech buněk na řádku R a sloupci C, pro které platí R1 <= R <= R2, C1 <= C <= C2. Pokud namísto čísla R2 resp. C2 bude pomlčka, nahrazuje tak maximální řádek resp. sloupec v tabulce.
- `[_,_]` - výběr celé tabulky.
- `[min]` - v již existujícím výběru buněk najde buňku s minimální numerickou hodnotou a výběr nastaví na ni.
- `[max]` - obdobně jako předchozí příkaz, ale najde buňku s maximální hodnotou.
- `[find STR]` - v již existujícím výběru buněk vybere první buňku, jejíž hodnota obsahuje podřetězec STR.
- `[_]` - obnoví výběr z dočasné proměnné (viz níže).

Příkazy pro úpravu struktury tabulky

- `irow` - vloží jeden prázdný řádek ~~nalevo od vybraných buněk~~ nad vybrané buňky.
- `arow` - přidá jeden prázdný řádek ~~napravo od vybraných buněk~~ pod vybrané buňky.
- `drow` - odstraní vybrané řádky.
- `icol` - vloží jeden prázdný sloupec nalevo od vybraných buněk.
- `acol` - přidá jeden prázdný sloupec napravo od vybraných buněk.
- `dcol` - odstraní vybrané sloupce.

Pozn. vložené řádky a sloupce mají účinek nad rámec vybraných buněk, přidávají nebo odebírají sloupce nebo řádky celé tabulky.

Příkazy pro úpravu obsahu buněk

- `set STR` - nastaví hodnotu buňky na řetězec STR. Řetězec STR může být ohraničen uvozovkami a může obsahovat speciální znaky uvozené lomítkem (viz formát tabulky)
- `clear` - obsah vybraných buněk bude odstraněn (buňky budou mít prázdný obsah)
- `swap [R,C]` - vymění obsah vybrané buňky s buňkou na řádku R a sloupci C
- `sum [R,C]` - do buňky na řádku R a sloupci C uloží součet hodnot vybraných buněk (odpovídající formátu %g u printf). Vybrané buňky neobsahující číslo budou ignorovány (jako by vybrány nebyly).
- `avg [R,C]` - stejné jako sum, ale ukládá se aritmetický průměr z vybraných buněk
- `count [R,C]` - stejné jako sum, ale ukládá se počet neprázdných buněk z vybraných buněk
- `len [R,C]` - do buňky na řádku R a sloupci C uloží délku řetězce aktuálně vybrané buňky

Příkazy pro práci s dočasnými proměnnými

Tabulkový procesor umožnujě pracovat s 10 tzv. dočasnými proměnnými identifikovanými jako _0 až _9. Dočasné proměnné mají po spuštění programu prázdnou hodnotu. Jedna dočasná proměnná identikovaná jako podtržítko _ je určena pro výběr buněk (tj. pamatuje si, které buňky byly označeny).

- `def _X` - hodnota aktuální buňky bude nastavena do dočasné proměnné X (kde X může být hodnota 0 až 9)
- `use _X` - aktuální buňka bude nastavena na hodnotu z dočasné proměnné X (kde X identifikuje dočasnou proměnnou _0 až _9)
- `inc _X` - numerická hodnota v dočasné proměnné bude zvětšena o 1. Pokud dočasná proměnná neobsahuje číslo, bude výsledná hodnota proměnné nastavená na 1.
- `[set]` - nastaví aktuální výběr buněk do dočasné proměnné _ (ukládá se pouze, které buňky jsou vybrány, nikoliv jejich obsah)

Příkazy pro řízení sekvence příkazů

Následující příkazy jsou volitelné a slouží pouze pro experimentování, inispiraci a pobavení:

- `goto +N` - přeskočí N následujících příkazů v sekvenci příkazů (pozn. `goto +1` je příkaz bez efektu)
- `goto -N` - vrátí se v sekvenci příkazů o N příkazů zpět (pozn. `goto -0` je zacyklení programu bez efektu).
- `iszero _X +-N` - přeskočí (v případě +N) nebo se vrátí (v případě -N) o N příkazů v sekvenci, ovšem pouze v případě, že hodnota dočasné proměnné _X (0 <= X <= 9) je rovna 0.

- `sub _X _Y` - od hodnoty v dočasné proměnné _X odečte hodnotu proměnné _Y. Výsledek uloží do proměnné _X.

Implementujte detektor zacyklení příkazů.

Výstup

Výstupní tabulka bude upravena podle následujících pravidel:

- Každá buňka bude oddělena prvním znakem ze sady DELIM.
- ~~Pokud buňka obsahuje uvozovku nebo znak, který je shodný s jedním z oddělovačů, bude ohraničen uvozovkami.~~
- ~~Pokud buňka obsahuje znaky, které by byly chápány jako speciální s ohledem na formát tabulky, budou tyto uvozeny zpětným lomítkem.~~
- Program nesmí měnit význam jednotlivých znaků v buňkách.
- Pokud text buňky obsahuje uvozovky nebo symbol oddělovače, bude celý text buňky ohraničen uvozovkami.
  - např. dvě buňky s mezerou jako oddělovačem, v C jako inicializátor pole `{"1", "hello world"}` bude vytištěn jako jeden řádek `1 "hello world"`
- Pokud text buňky obsahuje znak uvozovek nebo zpětné lomítko, tyto budou uvozeny zpětným lomítkem.
  - např. buňka s textem: He said: "Wow!" bude převedena na: "He said: \"Wow!\""
- Každý řádek tabulky bude obsahovat stejný počet buněk. Tabulka bude obsahovat všechny buňky, které jsou neprázdné. Tabulka nebude obsahovat žádný prázdný poslední sloupec.

Implementační detaily

- Maximální počet příkazů je 1000.
- Maximální délka jednoho příkazu je 1000.
- Žádné číslo příkazu identifikující řádek nebo sloupec nesmí být menší než 1.
- Pokud nějaký řádek obsahuje více buněk než jiný řádek tabulky, budou jiné řádky tabulky doplněny (zprava) o prázdné buňky tak, aby byla tabulka zarovnaná, tj. každý řádek obsahoval stejný počet buněk. Zarovnání řádků se provede ještě před aplikací prvního příkazu.

Tipy pro implementaci

- Implementujte funkci pro načtení řetězce s možností ohraničení uvozovkami a obsahující speciální znaky uvozené zpětným lomítkem. Implementujte funkci pro výpis takového řetězce odpovídající výstupnímu formátu tabulky.
- Definujte si datové typy pro buňku, řádek a tabulku.
- Implementujte funkce pro načtení tabulky ze souboru do paměti. Implementujte funkci pro zarovnání tabulky. Implementujte funkci pro výpis tabulky (na stdout pro účely vývoje a ladění, bez úpravy původního souboru).
- Definujte datový typ pro obecný příkaz (operace a parametry příkazu). Implementujte funkce pro převod textově definovaného příkazu do vámi definované struktury. Implementujte funkci pro převod sekvence textově definovaných příkazů na sekvenci příkazů.
- Definujte datový typ pro výběr buněk. Definujte dočasné proměnné `_`, `_0` až `_9`.
- Implementujte funkci výběru jednotlivých buněk. Implementujte funkce odpovídající vybraným příkazům zpracování tabulky.

- Implementujte funkce ostatních způsobů výběru. Implementujte funkce implementující ostatní příkazy.
- Implementujte funkci interpretu, který postupně vykonává jednotlivé příkazy na zadané sekvenci příkazů.
- Průběžně laďte, zkoumejte chování na různých situacích. Ověřujte korektní práci s pamětí pomocí nástroje valgrind.
- Implementujte řídicí příkazy. Experimujte s různými sekvencemi příkazů. Hledejte sekvence příkazů, které např.
    - přidají sloupec s očíslovanými řádky,
    - zduplikují tabulku,
    - vymažou všechny řádky kromě prvního, který obsahuje vybraný řetězec,
    - seřadí tabulku podle délky řetězce v prvním sloupci, atd.

Neočekávané chování

Na chyby za běhu programu reagujte obvyklým způsobem: Na neočekávaná vstupní data, formát vstupních dat nebo chyby při volání funkcí reagujte přerušením programu se stručným a výstižným chybovým hlášením na příslušný výstup a odpovídajícím návratovým kódem. Hlášení budou v kódování ASCII česky nebo anglicky.

Příklady vstupů a výstupů

```
$ cat tab.txt
Bill Gates 15000000
Franta Odvedle 15
Pepik Ajtak 42
```

Apostrofy v následujícím příkladu spuštění programu způsobí, že řetězec mezi nimi bude zadán jako jediný argument. Původní příklad obsahoval selekci buněk `1,_;irow`. U příkazů pro úpravy struktury tabulky nebudou tyto případy testovány. Budou testovány pouze u výběru jednotlivých buněk.

```
$ ./sps '[1,1];irow;[1,1];set Jmeno;[1,2];set Prijmeni;[1,3];set Plat' tab.txt
$ cat tab.txt
Jmeno Prijmeni Plat
Bill Gates 15000000
Franta Odvedle 15
Pepik Ajtak 42
$ ./sps '[_,_];[max];def _0;[2,3,-,3];use _0' tab.txt
$ cat tab.txt
Jmeno Prijmeni Plat
Bill Gates 15000000
Franta Odvedle 15000000
Pepik Ajtak 15000000
```

Poznámka: Následující očíslování lze dosáhnout jenodušeji. Zde pouze příklad pro aplikaci příkazu nad jednotlivými buňkami: swap se bude aplikovat opakovaně pro každou vybranou buňku, což má vedlejší efekt posunutí buněk o jedno doprava:

```
$ ./sps '[1,3];acol;[1,_];swap [1,4];inc _0;[2,_];swap [2,4];[2,1];use _0;inc
_0;[3,_];swap [3,4];[3,1];use _0;inc _0;[4,_];swap [4,4];[4,1];use _0' tab.txt
$ cat tab.txt
```

```
 Jmeno Prijmeni Plat
1 Bill Gates 15000000
2 Franta Odvedle 15000000
3 Pepik Ajtak 15000000
```

Hodnocení

Na výsledném hodnocení mají hlavní vliv následující faktory:

1. přeložitelnost zdrojového souboru,
    1. zdrojový soubor musí být přeložitelný a běžet na GNU/Linux, například na serveru merlin.fit.vutbr.cz (pozn. Kód má být napsán v jazyku C podle standardu ISO C 99, který má fungovat stejně na všech platformách. Jestliže kód nebude fungovat na stroji merlin, nebude fungovat na spoustě dalších platformách).
2. dekompozice problému na podproblémy (vhodné funkce, vhodná délka funkcí a parametry funkcí),
3. správná volba datových typů, tvorba nových typů,
4. správná funkcionalita:
    1. s možným omezením na velikost tabulky (lépe se soustředit na implementaci funkcionality nežli práce s neomezenou tabulkou)
    2. výběr jedné buňky, jednoho řádku, jednoho sloupce
    3. příkazy pro úpravu struktury tabulky (u příkazů pro úpravy struktury tabulky budou hodnoceny pouze případy nad výběrem jediné buňky)
    4. příkazy pro hledání v tabulce, příkazy pro agregující výpočty (sum, avg)
    5. další příkazy
5. ošetření chybových stavů.

Prémiové hodnocení

Prémiové hodnocení je dobrovolné a lze za něj získat bonusové body. Podmínkou pro udělení prémiových bodů je úspěšná obhajoba projektu a výborné vypracování standardního zadání. Výsledné hodnocení je plně v kompetenci vyučujího, kterému bude projekt obhajován a který bude projekt hodnotit. Výše prémiových bodů závisí také na sofistikovanosti řešení. Prémiová implementace by měla zahrnovat příkazy pro řízení.